# SPYWOLF

## Security Audit Report

Completed on

**November 25, 2023**

# OVERVIEW

This audit has been prepared for **4BOND** to review the main aspects of the project to help investors make make an informative decision during their research process.

You will find a a summarized review of the following key points:

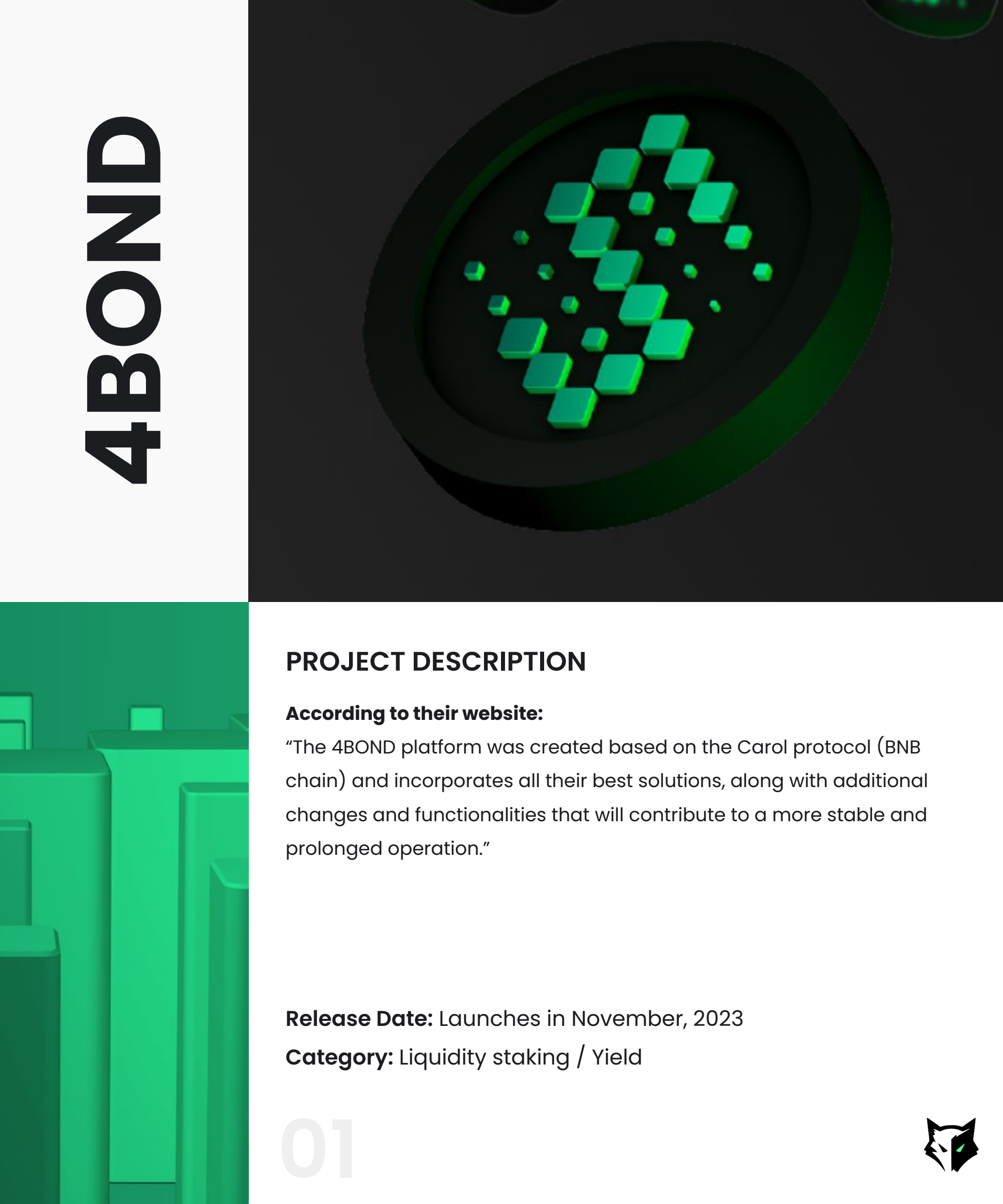
- ✔ Contract's source code
- ✔ Owners' wallets
- ✔ Tokenomics
- ✔ Team transparency and goals
- ✔ Website's age, code, security and UX
- ✔ Whitepaper and roadmap
- ✔ Social media & online presence

*The results of this audit are purely based on the team's evaluation and does not guarantee nor reflect the projects outcome and goal*

- SPYWOLF Team -

# TABLE OF CONTENTS

# 4BOND



## PROJECT DESCRIPTION

**According to their website:**

“The 4BOND platform was created based on the Carol protocol (BNB chain) and incorporates all their best solutions, along with additional changes and functionalities that will contribute to a more stable and prolonged operation.”

**Release Date:** Launches in November, 2023

**Category:** Liquidity staking / Yield

# CONTRACT 1 INFO (Main Contract )

Token Name
N/A

Symbol
N/A

Contract Address
0x63f5c33647c6185dbAbF37f527104f7970e9B1eE

Network
Binance Smart Chain

Language
Solidity

Deployment Date
Nov 24, 2023

Contract Type
Staking

Total Supply
N/A

Status
Not launched

# TAXES

Buy Tax
**Up to 15%**

Sell Tax
**none**

# Our Contract Review Process

The contract review process pays special attention to the following:

- Testing the smart contracts against both common and uncommon vulnerabilities
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

**Blockchain security tools used:**

- OpenZeppelin
- Mythril
- Solidity Compiler
- Hardhat

# TOKEN TRANSFERS STATS

| | |
|---|---|
| **Transfer Count** | N/A |
| **Uniq Senders** | N/A |
| **Uniq Receivers** | N/A |
| **Total Amount** | N/A |
| **Median Transfer Amount** | N/A |
| **Average Transfer Amount** | N/A |
| **First transfer date** | N/A |
| **Last transfer date** | N/A |
| **Days token transferred** | N/A |

# SMART CONTRACT STATS

| | |
|---|---|
| **Calls Count** | 5 |
| **External calls** | 5 |
| **Internal calls** | 0 |
| **Transactions count** | 5 |
| **Uniq Callers** | 3 |
| **Days contract called** | 1 |
| **Last transaction time** | 2023-11-24 13:11:10 UTC |
| **Created** | 2023-11-24 12:58:19 UTC |
| **Create TX** | 0xdbff0fa832303b664c7884210be59fc65798f2a5e54a3842fb88bbcfa418788c |
| **Creator** | 0xa97455c11a11bb83f6fe61c4be42a11661e2c48e |

# VULNERABILITY CHECK

| | |
|---|---|
| Design Logic | Passed |
| Compiler warnings. | Passed |
| Private user data leaks | Passed |
| Timestamp dependence | Passed |
| Integer overflow and underflow | Passed |
| Race conditions and reentrancy. Cross-function race conditions | Passed |
| Possible delays in data delivery | Passed |
| Oracle calls | Passed |
| Front running | Passed |
| DoS with Revert | Passed |
| DoS with block gas limit | Passed |
| Methods execution permissions | Passed |
| Economy model | Passed |
| Impact of the exchange rate on the logic | Passed |
| Malicious Event log | Passed |
| Scoping and declarations | Passed |
| Uninitialized storage pointers | Passed |
| Arithmetic accuracy | Passed |
| Cross-function race conditions | Passed |
| Safe Zeppelin module | Passed |
| Fallback function security | Passed |

# THREAT LEVELS

When performing smart contract audits, our specialists look for known vulnerabilities as well as logical and access control issues within the code. The exploitation of these issues by malicious actors may cause serious financial damage to projects that failed to get an audit in time. We categorize these vulnerabilities by the following levels:

## High Risk

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

## Medium Risk

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

## Low Risk

Issues on this level are minor details and warning that can remain unfixed.

## Informational

Information level is to offer suggestions for improvement of efficacy or security for features with a risk free factor.

# FOUND THREATS

## ⚠ Medium Risk

**Moderator can issue new bonds for free.**
New bonds can be issued only when the contract's current token balances are equal or higher than the new bond's issue token amount.

```
function influencerBond(address userAddr, uint256 tokensAmount, address upline) external onlyModerator {
    require(tokensAmount > 1e18, "Invalid tokens amount");
    require(userAddr != upline, "Upline can't be the same address as user address");
    require(users[userAddr].bondsNumber < Constants.BONDS_LIMIT, "User have reached bonds limit");
    require(IERC20(TOKEN_ADDRESS).balanceOf(address(this)) >= tokensAmount, "Insufficient token balance");
    if (upline == address(0x0)) {
        upline = DEFAULT_UPLINE;
    }
    if (users[userAddr].upline == address(0x0)) {
        users[userAddr].upline = upline;
        if (users[userAddr].lastActionTime == 0) {
        users[userAddr].lastActionTime = block.timestamp;
        }

        users[upline].referrals.push(userAddr);
        for (uint256 i = 0; i < REFERRAL_DEPTH; i++) {
        users[upline].refsNumber[i]++;
        upline = users[upline].upline;

        if (upline == address(0x0)) {
            break;
        }
        }
    }

    users[userAddr].balance+= tokensAmount * 5 / 100;
    uint256 ethAmount = getETHAmount(tokensAmount * 95 / 100);
    uint8 bondIdx = newBond(userAddr, 4, ethAmount, 0);

    CAROLToken(TOKEN_ADDRESS).burn(tokensAmount);

    emit Events.NewBond(
        userAddr, 4, bondIdx, ethAmount, tokensAmount * 95 / 100, false, block.timestamp
    );
}
```

- Recommendation:
  - No one should be able to issue new bonds for free.

# Informational

**Owner can activate/deactivate bond types (1, 2 and 3), which are for 20 days, 10 days and 5 days periods respectively.**
Bonds 0 and 4 (30 days and 100 days) cannot be influenced by owner
Every bond type have different ROI and freeze periods.

```
function activateBondType(uint8 bondType) external onlyOwner {
require(bondType > 0 && bondType < 4, "Invalid bond type");

    BOND_ACTIVATIONS[bondType] = true;
}

function deactivateBondType(uint8 bondType) external onlyOwner {
require(bondType > 0 && bondType < 4, "Invalid bond type");

BOND_ACTIVATIONS[bondType] = false;
}

int256[5] public BOND_FREEZE_PERIODS = [
    30 days,
    20 days,
    10 days,
    5 days,
    100 days
];
uint256[5] public BOND_FREEZE_PERCENTS = [
    3000,
    2000,
    1000,
    500,
    0
];
bool[5] public BOND_ACTIVATIONS = [
    true,
    false,
    false,
    false,
    false
];
```

# Informational

**There is 5% fee for bonds buy/staking that goes to the project's owner.** There is additional tax from 3% up to 10% (depending on users referrals and how much new capital they brought to the ecosystem) which goes towards referrals rewards.

```
uint256[] public REFERRAL_LEVELS_PERCENTS = [300, 400, 500, 600, 700, 800, 900, 1000];
uint256[] public REFERRAL_LEVELS_MILESTONES = [0, 10 ether, 30 ether, 100 ether,
350 ether, 700 ether, 1800 ether, 5500 ether];

function buy(address upline, uint8 bondType) external payable whenNotPaused {
.....................
    uint256 refReward = distributeRefPayout(user, msg.value, isNewUser);
    uint256 adminFee = msg.value / 20;
    payable(owner()).transfer(adminFee);

    newBond(msg.sender, bondType, msg.value, msg.value - adminFee - refReward);
}

function stake(uint8 bondIdx) external payable {
.....................
    uint256 refReward = distributeRefPayout(user, msg.value, false);
    uint256 adminFee = msg.value / 20;
    payable(owner()).transfer(adminFee);

    uint256 tokensAmount = getTokensAmount(ethAmount);
    ethAmount = msg.value - refReward - adminFee;
    uint256 liquidityTokensAmount = getTokensAmount(ethAmount);
.....................
}
```

# Informational

**When users set address they buy with for referral or address(0) or address that is not participating in the project yet (address with 0 bonds), the user becomes referral to default address assigned by project owner.**
Referral rewards can go from 3% up to 10% from user's deposited value based on how many total funds users collected from previous referrals.

```
function buy(address upline, uint8 bondType) external payable whenNotPaused {
  ......................
    bool isNewUser = false;
    Models.User storage user = users[msg.sender];
    if (user.upline == address(0)) {
      isNewUser = true;
      if (upline == address(0) || upline == msg.sender || users[upline].bondsNumber == 0) {
        upline = DEFAULT_UPLINE;
      }
      user.upline = upline;

      if (upline != DEFAULT_UPLINE) {
        users[upline].referrals.push(msg.sender);
      }

      emit Events.NewUser(
        msg.sender, upline, block.timestamp
      );
    }
.......................
}
```

# Informational

**Owner can pause new buys.**

```
modifier whenNotPaused() {
    _requireNotPaused();
    _;
}

function _requireNotPaused() internal view virtual {
    require(!paused(), "Pausable: paused");
}

function pause() external onlyOwner {
    _pause();
}

function _pause() internal virtual whenNotPaused {
    _paused = true;
    emit Paused(_msgSender());
}

function buy(address upline, uint8 bondType) external payable whenNotPaused {
    require(!msg.sender.isContract(), "Buy: user can't be a contract");
    require(bondType < 4 && BOND_ACTIVATIONS[bondType], "Buy: invalid bond type");
    require(users[msg.sender].bondsNumber < Constants.BONDS_LIMIT, "Buy: you have reached bonds limit");
    require(msg.value >= Constants.MIN_BOND_ETH, "Buy: min buy amount is 0.01 BNB");

    bool isNewUser = false;
    Models.User storage user = users[msg.sender];
    if (user.upline == address(0)) {
      isNewUser = true;
      if (upline == address(0) || upline == msg.sender || users[upline].bondsNumber == 0) {
        upline = DEFAULT_UPLINE;
      }
      user.upline = upline;

      if (upline != DEFAULT_UPLINE) {
        users[upline].referrals.push(msg.sender);
      }

      emit Events.NewUser(
        msg.sender, upline, block.timestamp
      );
    }

    uint256 refReward = distributeRefPayout(user, msg.value, isNewUser);
    uint256 adminFee = msg.value / 20;
    payable(owner()).transfer(adminFee);

    newBond(msg.sender, bondType, msg.value, msg.value - adminFee - refReward);
}
```

# Informational

**Users can claim for another address.**
Users will send their rewards to the selected address.

```
function claim(uint256 tokensAmount, address receiver) external {
    require(userBalance(msg.sender) >= tokensAmount, "Claim: insufficient balance");

    collect(msg.sender);
    Models.User storage user = users[msg.sender];
    require(user.balance >= tokensAmount, "Claim: insufficient balance");

    user.balance-= tokensAmount;
    user.lastActionTime = block.timestamp;

    if (receiver == address(0x0)) {
        receiver = msg.sender;
    }
    CAROLToken(TOKEN_ADDRESS).mint(receiver, tokensAmount);

    emit Events.Claim(
        msg.sender, receiver, tokensAmount, block.timestamp
    );
}
```

**When address different than address(0) is selected user will forfeit their rewards in favour of the selected address.*

# Informational

***Users can rebond for another address.**
Receiver address must be already registered.

```
function rebond(uint256 tokensAmount, address receiver) external {
    require(!receiver.isContract(), "Rebond: user can't be a contract");
    if (receiver == address(0x0)) {
      receiver = msg.sender;
    }
    require(users[receiver].lastActionTime > 0, "Rebond: receiver doesn't exist");

    require(users[receiver].bondsNumber < Constants.BONDS_LIMIT, "Rebond: receiver have reached bonds limit");
    require(tokensAmount >= Constants.MIN_BOND_TOKENS, "Rebond: min rebond amount is 100 CAROL");
    require(userBalance(msg.sender) >= tokensAmount, "Rebond: insufficient balance");

    collect(msg.sender);
    Models.User storage user = users[msg.sender];
    require(user.balance >= tokensAmount, "Rebond: insufficient balance");

    user.balance-= tokensAmount;

    uint256 ethAmount = getETHAmount(tokensAmount);
    uint8 bondIdx = newBond(receiver, 0, ethAmount, 0);

    emit Events.ReBond(
      receiver, bondIdx, ethAmount, tokensAmount, block.timestamp
    );
}
```

**When address different than address(0) is selected user will forfeit their rebond in favour of the selected address.*

# Informational

**Owner can change PRICE_BALANCER_PERCENT's value.**
This variable is responsible for the current sell ratio price.

```
function changePriceBalancerPercent(uint256 percent) external onlyOwner {
    require(percent >= 0 && percent <= 20000, "Invalid percent amount (0 - 20000)");
    PRICE_BALANCER_PERCENT = percent;
}

function sell(uint256 tokensAmount) external {
.............................
    if (PRICE_BALANCER_PERCENT > 0) {
        (uint256 ethReserved, ) = getTokenLiquidity();
        uint256 liquidity = ERC20(LP_TOKEN_ADDRESS).totalSupply()
        * ethAmount
        * PRICE_BALANCER_PERCENT
        / Constants.PERCENTS_DIVIDER
        / ethReserved;

        ERC20(LP_TOKEN_ADDRESS).approve(
        UNISWAP_ROUTER_ADDRESS,
        liquidity
        );

        (, uint256 amountETH) = IUniswapV2Router01(UNISWAP_ROUTER_ADDRESS).removeLiquidityETH(
        TOKEN_ADDRESS,
        liquidity,
        0,
        0,
        address(this),
        block.timestamp + 5 minutes
        );

        path[0] = Constants.WRAPPED_ETH;
        path[1] = TOKEN_ADDRESS;
        amounts = IUniswapV2Router01(UNISWAP_ROUTER_ADDRESS).swapExactETHForTokens {value: amountETH} (
        0,
        path,
        address(this),
        block.timestamp + 5 minutes
        );
    }
.............................

}
```

# Informational

**Owner can add/remove moderators.**

```
function addModerator(address moderator) external onlyOwner {
    moderators[moderator] = true;
}

function removeModerator(address moderator) external onlyOwner {
    moderators[moderator] = false;
}
```

**Owner can withdraw any tokens from the contract except the native FUD token.**
When this function is present, in cases tokens are sent into the contract by mistake or purposefully, contract's owner can retrieve them.

```
function retrieveERC20(address tokenAddress, uint256 amount) external onlyOwner {
    require(tokenAddress != TOKEN_ADDRESS, "You can't retrieve project token");

    if (amount == 0) {
      amount = IERC20(tokenAddress).balanceOf(address(this));
    }

    IERC20(tokenAddress).transfer(owner(), amount);
}
```

**Anyone can set their refBackPercent up to 100%.**
For more information about ref back check project's documents page:
https://4bond.gitbook.io/4bond-bonding-and-staking-platform/referral-program/refback

```
function setRefbackPercent(uint256 refbackPercent) external {
    require(refbackPercent <= 10000, "Invalid percent amount (0 - 10000)");

    Models.User storage user = users[msg.sender];

    if (user.bondsNumber > 0) {
      user.refbackPercent = refbackPercent;
    }
}
```

## RECOMMENDATIONS FOR

# GOOD PRACTICES

1. Consider fundamental tradeoffs
2. Be attentive to blockchain properties
3. Ensure careful rollouts
4. Keep contracts simple
5. Stay up to date and track development

## 4BOND

### GOOD PRACTICES FOUND

- ✔ The owner cannot set a transaction limit

# CONTRACT 2 INFO (Token)

Token Name
FUD

Symbol
FUD

Contract Address
0xDe0F92B659367f55085Fb4EbE198Cf4069ecDC1c

Network
Binance Smart Chain

Language
Solidity

Deployment Date
Nov 24, 2023

Contract Type
Token

Total Supply
10,000,000

Status
Not launched

## TAXES

Buy Tax
**N/A**

Sell Tax
**N/A**

# Our Contract Review Process

The contract review process pays special attention to the following:

- Testing the smart contracts against both common and uncommon vulnerabilities
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

**Blockchain security tools used:**

- OpenZeppelin
- Mythril
- Solidity Compiler
- Hardhat

# FOUND THREATS

## ⚠ High Risk

No high risk-level threats found in this contract.

## ⚠ Medium Risk

No medium risk-level threats found in this contract.

## ⚠ Low Risk

No low risk-level threats found in this contract.

# Informational

Owner can enable/disable token buys.

```
function unlockBuy() external onlyOwner {
    buyLocked = false;
}

function lockBuy() external onlyOwner {
    buyLocked = true;
}

function _beforeTokenTransfer(address from, address to, uint256 ) internal view override {
    if (LP_TOKEN_ADDRESS == address(0) || !buyLocked) {
      return;
    }

    if (from == LP_TOKEN_ADDRESS || from == UNISWAP_ROUTER_ADDRESS) {
      require(
            to == mainContractAddress
        || to == UNISWAP_ROUTER_ADDRESS
        || to == LP_TOKEN_ADDRESS
        || to == address(0),
        "Transfer: only main contract can buy tokens"
      );
    }
}
```

## RECOMMENDATIONS FOR

# GOOD PRACTICES

1. Consider fundamental tradeoffs
2. Be attentive to blockchain properties
3. Ensure careful rollouts
4. Keep contracts simple
5. Stay up to date and track development

## 4BOND

### GOOD PRACTICES FOUND

- ✔ The owner cannot set a transaction limit

This is *ROI staking dapp with referral system that allows users to get up to 10% from each referral. When users choose to stake their capital (bonds/liquidity) they can earn up to 150% of their initial investment over time.

More information can be found in the project's documents page:
*https://4bond.gitbook.io/4bond-bonding-and-staking-platform/core-mechanics-and-functionality/bonding*

**ROI dapps are considered as high risk and can cause significant losses of capital.**
***DYOR before investing in any.**

**ROI - **R**eturn **O**f **I**nvestment*
**DYOR - **D**o **Y**our **O**wn **R**esearch*



TOKENOMICS

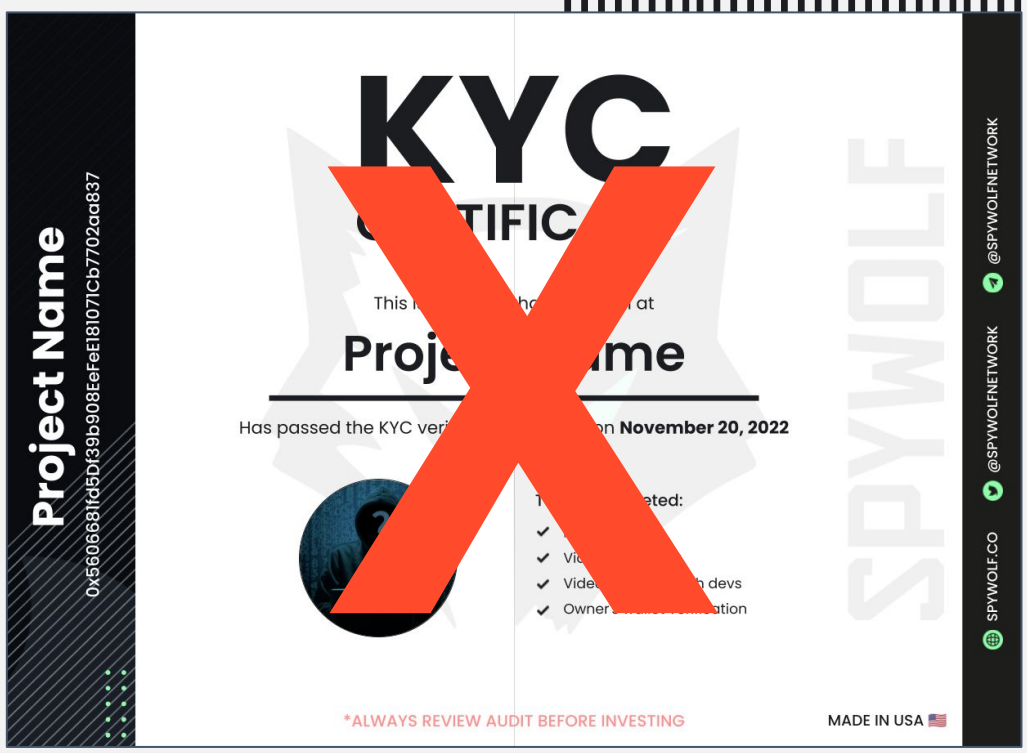
# THE TEAM

⚠️ The team is annonymous

## KYC INFORMATION

### No KYC

We recommend the team to get a KYC in order to ensure trust and transparency within the community.



WEBSITE

## Website URL
https://4bond.io/

## Domain Registry
https://www.namecheap.com/

## Domain Expiration
2024-11-10

## Technical SEO Test
Passed

## Security Test
Passed. SSL certificate present

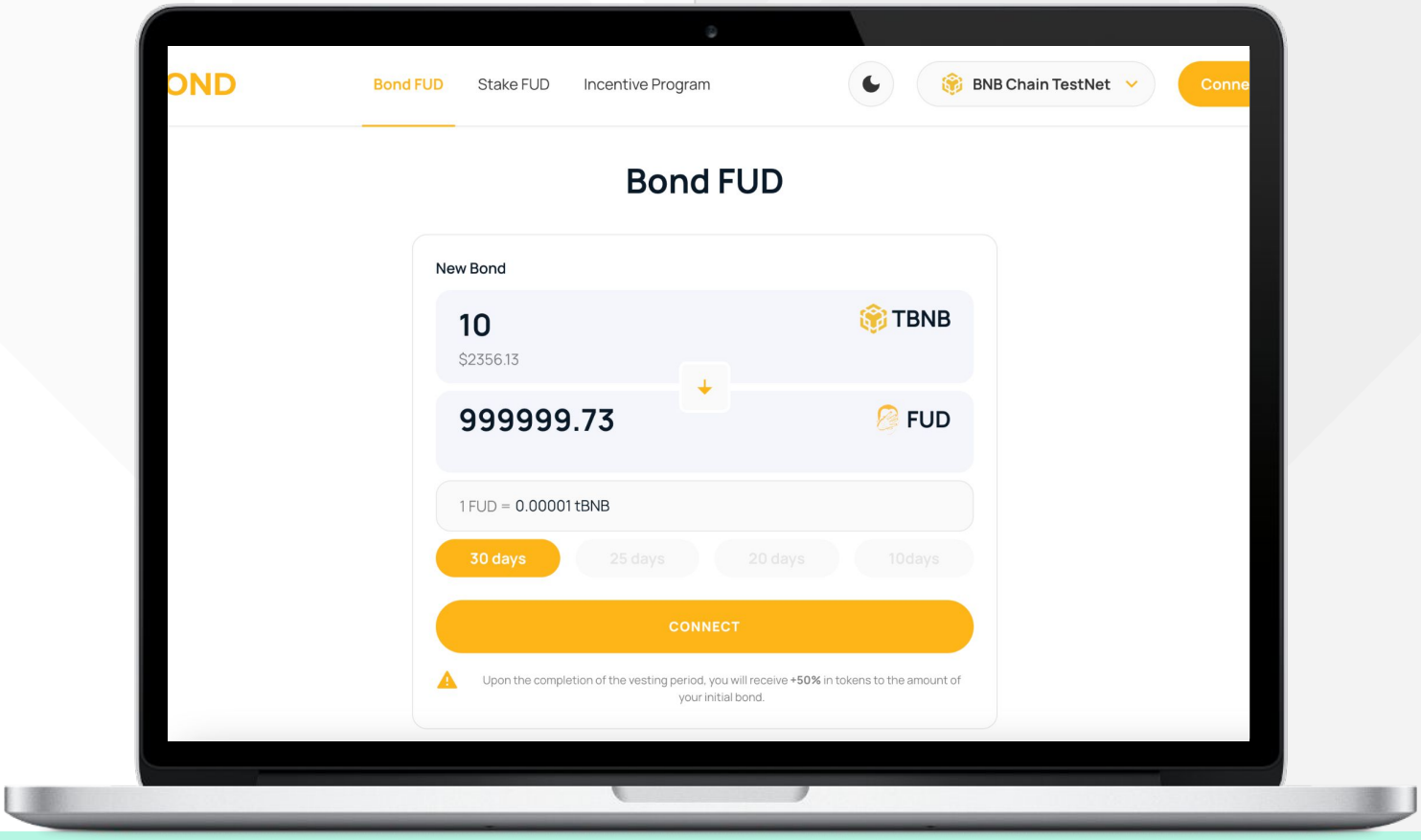
## Design
Very nice overall design with appropriate color scheme and graphics.

## Content
The information helps new investors understand what the product does right away. No grammar mistakes found
.

## Whitepaper
Well written and explanatory documents page.

## Roadmap
No

## Mobile-friendly?
Yes




# 4bond.io

# SOCIAL MEDIA

## & ONLINE PRESENCE

**ANALYSIS**

**Project's social media pages are under construction**


### Twitter

@4bond_io

- 30 followers
- New account

### Discord

- Not available

### Telegram

@FourBond_group

- 1 member
- New account

### Medium

- Not available

# ABOUT US

We are a growing crypto security agency offering audits, KYCs and consulting services for some of the top names in the crypto industry.

- **OVER 700 SUCCESSFUL CLIENTS**
- **MORE THAN 1000 SCAMS EXPOSED**
- **MILLIONS SAVED IN POTENTIAL FRAUD**
- **PARTNERSHIPS WITH TOP LAUNCHPADS, INFLUENCERS AND CRYPTO PROJECTS**
- **CONSTANTLY BUILDING TOOLS TO HELP INVESTORS DO BETTER RESEARCH**

SPYWOLF

CRYPTO SECURITY

Audits | KYCs | dApps
Contract Development

To hire us, reach out to
contact@spywolf.co or
t.me/joe_SpyWolf

## FIND US ONLINE

SPYWOLF.CO

@SPYWOLFNETWORK

@SPYWOLFNETWORK

# Disclaimer

This report shows findings based on our limited project analysis, following good industry practice from the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, overall social media and website presence and team transparency details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report.
While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

**DISCLAIMER:**
By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice.
No one shall have any right to rely on the report or its contents, and SpyWolf and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (SpyWolf) owe no duty of care towards you or any other person, nor does SpyWolf make any warranty or representation to any person on the accuracy or completeness of the report.
The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and SpyWolf hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, SpyWolf hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against SpyWolf, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts, website, social media and team.
No applications were reviewed for security. No product code has been reviewed.